Extending the ABC-Miner Bayesian Classification Algorithm

Khalid M. Salama and Alex A. Freitas

School of Computing, University of Kent,
Canterbury, CT2 7NF, UK
{kms39,A.A.Freitas}@kent.ac.uk

Abstract. ABC-Miner is a Bayesian classification algorithm based on the Ant Colony Optimization (ACO) meta-heuristic. The algorithm learns Bayesian network Augmented Naïve-Bayes (BAN) classifiers, where the class node is the parent of all the nodes representing the input variables. However, this assumes the existence of a dependency relationship between the class variable and *all* the input variables, and this relationship is a type of "causal" (rather than "effect") relationship, which restricts the flexibility of the algorithm to learn. In this paper, we propose ABC-Miner+, an extension to the ABC-Miner algorithm which is able to learn more flexible Bayesian network classifier structures, where it is not necessary to have a (direct) dependency relationship between the class variable and each of the input variables, and the dependency between the class and the input variables varies from "causal" to "effect" relationships. The produced model is the *Markov blanket* of the class variable. Empirical evaluations on UCI benchmark datasets show that our extended ABC-Miner+ outperforms its previous version in terms of predictive accuracy, model size and computational time.

Keywords: Ant Colony Optimization (ACO), Data Mining, Classification, Bayesian Network Classifiers.

1 Introduction

Ant Colony Optimization (ACO) is a meta-heuristic for solving combinatorial optimization problems, inspired by the observation of the behavior of biological ant colonies [6]. One of the fields in which ACO has been successfully applied is data mining, which involves finding hidden patterns and constructing analytical models from real-world datasets [20]. Classification is one of the widely studied data mining tasks, where the aim is to discover, from labeled cases (instances), a model that can be used to predict the class of unlabeled cases. While there are several types of classification methods, such as decision tree and rule induction, artificial neural networks and support vector machines [20], our focus is on Bayesian network (BN) classifiers.

BN classifiers model the (in)dependency-relationships between the input domain variables given the class variable by means of a probabilistic network [7], which is used to predict the class of a case by computing the class with the highest posterior probability given the case's predictor attribute values. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-hard [4], stochastic heuristic search algorithms - such as ACO – can be a good alternative to build high-quality models, in terms of predictive accuracy and network size, within an acceptable computational time. Developing ACO-based algorithms to learn BN classifiers is the research topic addressed in this work.

We have recently introduced ABC-Miner [19], as an Ant-based Bayesian Classification algorithm that learns the structure of a Bayesian network Augmented Naïve-Bayes (BAN), where the class node is the parent of all the input variables, and at most k parents are allowed for each variable in the network. The ABC-Miner algorithm showed predictive effectiveness compared to other Bayesian classification algorithms, namely: Naïve-Bayes, TAN and GBN [19].

In this paper, we propose ABC-Miner+, which extends our ABC-Miner algorithm to learn more flexible BN classifier structures, where it is not necessary to have a (direct) dependency relationship between the class variable and each of the input variables. In addition, ABC-Miner+ allows the dependency between the class and the input variables to vary from "causal" to "effect" relationships, where the class variable can be a parent or a child of an input node. The produced model is called the *Markov blanket* of the class variable. Empirical results on 18 UCI datasets show that ABC-Miner+ improves the performance of ABC-Miner by producing simpler (smaller) BN classifiers that have higher predictive accuracy in less computational time.

Note that we use the word "causal" in a loose sense in this work, simply to refer to a direction of the dependency relationship between two variables. The issue of whether or not Bayesian networks learned from observational data represent truly causal knowledge is controversial (depending on how we define causality) [15], and is out of the scope of this paper.

The rest of the paper is structured as follows. The next section gives some background on BN classifiers. We briefly review the previously introduced ABC-Miner algorithm in Section 3, to make this paper more self-contained. Our proposed extension, ABC-Miner+ is described in detail in Section 4. We describe our experimental methodology and show the results in Section 5. Finally, we conclude with some general remarks and future research in Section 6.

2 Bayesian Network Classifiers

Bayesian networks (BNs) are knowledge representation and reasoning tools that model probabilistic dependence and independence relationships amongst variables in a specific domain [5]. Learning a BN from a dataset (in which the attributes are referred to as variables) consists of two phases: learning the network structure, and then learning the parameters of the network. Parameter learning is relatively straightforward for any given BN structure with specified dependencies between variables. The task is to estimate a conditional probability table (CPT), one for each variable, by computing the relative frequencies of the variable with respect to its parents directly from the dataset. The CPT of variable X_i encodes the likelihood of each value of this variable given each combination of values of the parents of variable X_i in the network.

There are two paradigms for learning the structure of a BN. The first one is referred to as *CI-based* (Conditional Independence-based, or constraint-based) algorithms [8, 5], which suggests learning the BN structure by identifying the conditional independence relationships among the nodes, according to the concept of *d-separation*. The second paradigm views the BN as a structure that encodes the joint distribution of the attributes. Hence, the aim is to find the graph that best fits a given dataset in terms of maximizing the value of a scoring function, which led to the scoring-based algorithms [8, 5]. In the context of data mining, the scoring-based approach has been (overall) more popular and it is somewhat easier to be used than the CI-based approach, partly because the former views the problem as a well-defined optimization task, where various search and meta-heuristic techniques can be employed [3]. K2, MDL, KL, BDEu and several other scoring functions can be used for this task [8].

A recent, very comprehensive review on BN-learning approaches and issues is presented by Daly et al. in [5]. For further information about BNs, the reader is referred to [8].

While BNs should perform inference to answer probabilistic queries about any node(s) in the network, BN classifiers are a special kind of the probabilistic networks, which focus on answering queries about the probability of a specific node: the class attribute. Thus, the class node is treated as a special variable in the network. The purpose is to compute the probability of each value c in the class variable C given a case $\mathbf{x}$ (an instance of the input attributes $\mathbf{X}$) using classifier BNC, then label the case with the class having the highest probability, as in the following formulas:

$$C(\mathbf{x}) = \underset{\forall\ c \in C}{\arg\max}\, P(C = c|\mathbf{x} = x_1, x_2, ..., x_n, BNC), \tag{1}$$

$$\overbrace{P(C = c|\mathbf{x} = x_1, x_2, ..., x_n)}^{\text{posterior probability}} \propto \overbrace{P(C = c)}^{\text{proir probability}} \prod_{v=1}^{n} \overbrace{P(x_v|\mathbf{Parents}(X_v), BNC)}^{\text{likelihood}}, \tag{2}$$

where $C \in \mathbf{Parents}(X_v) \forall\ X_v \in \mathbf{X}$.

Naïve-Bayes is the simplest kind of BN classifiers; it has a network structure where the class node is the only parent node of all other nodes (input variables). This structure assumes that all attributes are independent of each other given the class. However, in many real-world application domains this assumption is not satisfied, and more sophisticated types of BN classifiers, which consider dependencies between the predictor attributes, can lead to higher predictive accuracy. This led to the development of a general type of BN classifiers called BN Augmented Naïve-Bayes (BAN).

In a BAN classifier, each node representing an input attribute not only has the class node as a parent, but it is also allowed to have other parent nodes which are also input attributes. Hence, the edges representing dependencies among input attributes can be regarded as a kind of BN, which justifies the name "BN-augmented" Naïve-Bayes. Usually, however, each node representing a input attribute is allowed to have a maximum number (k) of parents, in order to reduce computational time and reduce the chances of over-fitting the BN to the data, and in this case the algorithm is often referred to as a k-dependency BAN. Note that when the maximum number of parents k is set to 1, the BAN is usually referred to as a TAN (Tree-Augmented Naïve-Bayes), because in that type of classifier each node representing an input attribute can have at most one parent node (in addition to the class node), so that the dependencies among input attributes are represented as a tree.

Figure 1 illustrates the various kinds of the BN classifiers. Friedman et al. provided an excellent study of these algorithms in [7]. A comprehensive comparison of these various Bayesian classifiers by Cheng and Greiner is found in [3]. Surveys on improving Naïve-Bayes for classification are found in [10, 11].

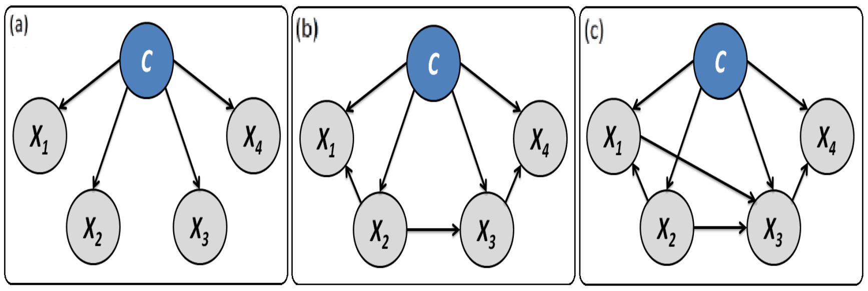

Fig. 1. Different types of BN classifiers: (a) Naïve-Bayes, (b) TAN, and (c) BAN.

3 An Overview of the ABC-Miner Algorithm

ACO algorithms have been successful in solving several combinatorial optimization problems, including classification rule discovery [12–14, 18, 17] and general purpose BN construction [2, 16, 21]. However, ABC-Miner, introduced by the authors in [19], is the first ACO algorithm that learns the structure of BAN classifiers [19].

In ABC-Miner, the decision components in the construction graph (which define the search space that an ant uses to construct a candidate solution) are all the edges of the form $X \rightarrow Y$ where $X \neq Y$ and X, Y belong to the set of input attributes. These edges represent the attribute dependencies in a constructed BN classifier – i.e., an edge $X \rightarrow Y$ means that the value of Y depends (probabilistically) on the value of X.

In order to build the structure of a BN classifier, the maximum number of parents for a node is typically specified by the user. However, the selection of the optimal number of parents (dependencies) that a variable in the network can have (in addition to having the class as a parent node) is automatically carried out in ABC-Miner [19]. To create a candidate solution, an ant starts with the network structure of Naïve-Bayes, where every variable has only the class variable as its parent. Then the ant expands that structure into a BAN structure by adding edges to the network. The selection of the edges is performed according to a probabilistic state transition formula that involves the pheromone amount and the heuristic function value – measured by the conditional mutual information [19] – of the edges. An edge is valid to be added to the BN classifier being constructed if its inclusion does not create a directed cycle and does not exceed the limit of k parents (chosen by the current ant).

After the ant adds a valid edge to the current candidate solution (BN classifier), all the invalid edges are eliminated from the construction graph. The ant keeps adding edges to the current solution until no valid edges are available. When the structure is finished, the CPT of each variable is computed, producing a complete BN classifier. Then the quality of the solution is evaluated and all the edges become available for constructing further candidate solutions. The ABC-Miner algorithm evaluates the quality of the candidate constructed BN classifier using a measure of predictive accuracy [19], since the goal is to build a BN only for predicting the value of a specific class attribute, unlike conventional BN learning algorithms whose scoring function does not distinguish between the input (predictor) and the class attributes.

4 The Proposed ABC-Miner+ Extension

The motivation behind our proposed extension is the following. As mentioned in the previous section, the structure of the BAN models constructed by ABC-Miner has two limitations. First, it assumes that the class variable has dependency relationships with *all* the input variables (the case's attributes), which means that the state of each input variable affects the posterior probability of the class values, and consequently the class prediction. This assumption is not necessarily valid in all applications domains. In some domains, some attributes are irrelevant, or at least not directly related, to the prediction of the target class. Including these irrelevant attributes in the computation of the posterior probability of the class values, according to Equation 2, can be disadvantageous, and may lead to incorrect predictions.

Second, in the BAN classifier constructed by ABC-Miner, the relationship between the class and all the input variables is always a type of "causal" relationship, that is, the class variable can only be a parent of an input variable. Such a property limits the flexibility of the algorithm to learn. Nonetheless, in real-world domains, some input variables are "causes" (parents) of the class variable, whereas others are "effects" (children) of the same class variable. For example, in a cancer diagnosis domain, the state of the *smoker* variable can be considered a cause of the state of the *Cancer* class variable, while the state of the *X-Ray* variable can be considered an effect of the class variable.

Accordingly, we propose ABC-Miner+, which extends the ABC-Miner algorithm to learn more flexible BN classifier structures, where it is not necessary to have a (direct) dependency relationship between the class variable and each of the input variables. This means that an input variable may not have a direct connection (edge) to the class node in the network, or an input variable may not even be presented in the network. In this case, our proposed ABC-Miner+ performs an embedded feature selection during the construction of the BN classifier. In addition, ABC-Miner+ allows the type of dependency (edge) between the class and the input variables to vary from “causal” to “effect” relationships, where the class variable can be a parent or a child of an input node.

The advantage of allowing this kind of edges in the BN model is the possibility of capturing new conditional (in)dependency-relationships. For example, if X and Y are input variables that are unconditionally independent of the class variable C, then X and Y should be parents to C. This kind of (in)dependency-relationship cannot be modeled by a BAN structure. Such a flexible BN classifier structure should better represent the dependency-relationships between the input variables, with respect to the class variable, and lead to higher classification accuracy. The produced model is the *Markov blanket* of the class variable, which consists of the class node's parents, the class node's children, and the parents of class node's children. Algorithm 1 shows the outline of ABC-Miner+.

Algorithm 1 Pseudo-code of ABC-Miner+.

```
Begin
$BNC_{final} = \phi; STR_{bsf} = \phi;$
$sets = trainingSet.Split();$ /* split training set into learning and validation sets */
$learningSet = sets[0];\ validationSet = sets[1];$
$Initialize();\ t = 1;$
repeat
    $STR_{tbest} = \phi;$ /* an empty network structure */
    for $i = 1 \rightarrow$ colony_size do
        $STR_i = FindRelationshipTypes(ant_i);$ /* create a candidate solution */
        $LearnParameters(STR_i, learningSet);$
        if $Quality(STR_i, validationSet) > Quality(STR_{tbest}, validationSet)$ then
            $STR_{tbest} = STR_i;$
        end if
    end for
    $UpdatePheromone(STR_{tbest});$
    if $Quality(STR_{tbest}, validationSet) > Quality(STR_{bsf}, validationSet)$ then
        $STR_{bsf} = STR_{tbest};$
    end if $t = t + 1;$
until $t =$ max_iterations or $Convergence();$
$STR_{final} = PerformLocalSearch(STR_{bsf});$
$BNC_{final} = ExecuteABCMiner(STR_{final});$ /* extend the final structure */
return $BNC_{final};$
End
```

ABC-Miner+ executes in two sequential phases. First, it finds the dependency relationship type between the class variable and each of the input variables. Second, it finds the dependency relationships among the input variables. Each step is considered a different ACO procedure and has a different construction graph. In the first phase, the product is a BN structure STR that defines the edges only between the input variables and the class variable, if any exists, and does not contain edges between the input variables. The decision components in the construction graph of the first phase are a set of relationship types between pairs of variables (attributes). More precisely, there are three decision components for each variable, representing the various relationship types that the variable can have with the class node: 1) "cause", where the class is a parent of the variable, 2) "effect", where the class is a child of the variable, and 3) "none", where there is no relationship between the class and the variable in the network, so that the algorithm can perform variable (feature) selection.

The idea is to find the best edges between the class and the input variables. Each ant_i constructs a candidate solution (BN structure), via the FindRelationshipTypes() method, as follows. For each input variable, ant_i probabilistically selects a relationship type, according to the pheromone amounts currently associated to the decision components in the construction graph, and adds its corresponding directed edge to the current candidate BN structure STR_i. Note that no edge is added between a variable and the class node in the case of selecting the "none" relationship type. The method returns a complete candidate solution (i.e. a BN structure where all the relationships between the class and the input variables are defined) before the BN parameters are learnt, and then the quality of the solution is evaluated. The algorithm learns the BN parameters using the learning set (containing 70% of the training cases), while the quality is evaluated on a validation set (containing the remaining 30% of the training cases), in order to try avoiding over-fitting to the training set. The quality of a candidate solution is evaluated as a BN classifier, using classification accuracy (Equation 3), before the iteration-best STR_{tbest} is used to perform pheromone update. The best-so-far STR_{bsf} structure undergoes local search, and the optimized STR_{final} structure is produced to be used in the next phase.

$$Accuracy = \frac{|Correctly\_Classified\_Cases|}{|Validation\_Set|} \tag{3}$$

In the second phase, the best constructed and optimized STR_{final} structure of the BN is extended to a complete class Markov blanket, by finding the dependency relationships among the input variables. Note that the BN structure discovered in the first step contains no edges between the input variables. To include this type of edge in the network structure, we execute the original ABC-Miner algorithm in this phase. However, in the context of ABC-Miner+, the solution creation procedure starts with the STR_{final} structure constructed in the previous phase, rather than a Naïve-Bayes structure as in the original ABC-Miner algorithm. The process of extending the BN structure to a candidate class Markov blanket, which takes place in the second phase of ABC-Miner+, is described in Algorithm 2. The algorithm shows just the process for each ant, for the sake of simplicity, since the overall pseudo-code of ABC-Miner has been already described in [19].

Algorithm 2 ABC-Miner+'s Second Phase: Ant Solution Creation Procedure.

```
Begin CreateSolution(ant) /* initialize the candidate Markov blanket solution with
the structure of $STR_{final}$ discovered in phase 1 */
$MB \leftarrow STR_{final}$ ;
$k = ant.SelectMaxParents()$;
while $GetValidEdges() \neq \phi$ do
    $\{i \rightarrow j\} = ant.SelectEdgeProbablistically()$;
    $MB = MB \cup \{i \rightarrow j\}$;
    $RemoveInvalidEdges(MB, k)$;
end while
$MB.LearnParameters(learningSet)$;
return $MB$;
End
```

The execution of the procedure shown in Algorithm 2 is more efficient than its corresponding solution creation procedure in the original ABC-Miner algorithm in several ways. First, the search space of this procedure in the context of ABC-Miner+ is smaller than the search space in context of the original ABC-Miner. The reason is that, in ABC-Miner, the initial structure is the Naïve-Bayes' structure, where all the input variables are children of the class variable, so all the candidate edges between the input variables are available for selection by an ant (i.e. any variable can be a parent to any other variable). On the other hand, in ABC-Miner+, the initial structure has some input variables as parents of the class variable, and others are not even related to the class variable. In this case, the candidate edges available for selection to be added to the network are only the edges that satisfy two conditions, namely: the edge is connecting two input variables (rather than connecting an input variable to the class), and the edge is pointing to a child node of the class node. The algorithm does not consider adding edges between the class variable's parents because these edges do not affect the predictions (posterior probability calculation) of the BN classifier.

Second, in the Markov blanket produced by ABC-Miner+, the size of the CPT for the variables that do not have the class variable as parent is relatively smaller compared to the CPT of the BN classifiers produced by ABC-Miner, where the class node has to be a parent to all the variables, besides their other parents. Smaller CPT size means less computational time.

Note that in the case of a Markov blanket (MB) classifier, both causal (parent) variables, and the effect (child) variables of the class variable are used to compute the posterior probability $P(c|\mathbf{x})$ of class c given case $\mathbf{x}$, along with the parents of the class node's children, according to the following formula:

$$P(c|\mathbf{x}) \propto P(c|\mathbf{Parents}(c)) \prod_{v \in m} P(x_v|\mathbf{Parents}(X_v), MB), \tag{4}$$

where m is the set of the input variables that have the class variable as parent.

5 Experiments and Computational Results

We compared the performance of our proposed ABC-Miner+ with two other BN classifier algorithms. The first one is basically the first phase of the ABC-Miner+ algorithm, where the BN classifiers produced have a structure only with the relationships between the class and the input variables, without discovering the dependency relationships among the input variables. The algorithm is denoted as ABC. The second algorithm is the original ABC-Miner, which produces BN classifiers with the structure of a BAN (where the class variable is a parent of all the input variables). The evaluation criteria consist of the following three types of performance measures: predictive accuracy (in general the most important criterion in the classification task of data mining), model size (measured by the total number of the edges in the network), and the running time.

The performance of the algorithms was evaluated using 18 public-domain datasets from the University of California at Irvine (UCI) dataset repository. The main characteristics of the datasets are shown in the URL in [1]. The experiments were carried out using the well-known *stratified* 10-fold cross validation procedure [20]. Since the ACO algorithms are stochastic, we run each 10 times – using a different random seed to initialize the search each time – for each of the 10 iterations of the cross-validation procedure. As for the parameter configurations, we set `colony_size` to 10 and `max_iterations` to 1000. Note than in the case of ABC-Miner+, each phase is allocated half of the total maximum number of iterations (i.e. 500 iterations in our experiments).

Table 1 shows the experimental results of the algorithms in three parts, one for each type of performance measure: predictive accuracy, model size, and running time. The entries in the table represent the mean values obtained by 10-fold cross validation. For each performance measure, the best result for each dataset is underlined.

In terms of predictive accuracy, the extended ABC-Miner+ algorithm obtained the best results in 14 out of 18 datasets, while ABC-Miner and ABC obtained the best results in 5 and 3 datasets, respectively. ABC-Miner+ outperformed ABC-Miner in 13 datasets plus 1 tie, while ABC outperformed ABC-Miner in 3 datasets. It is interesting to notice that ABC, which is only the first phase of ABC-Miner+ can find the best BN classification model in some datasets, and in those datasets the second phase of ABC-Miner+ does not improve its performance. This can be noticed in datasets `hayes`, `monk` and `pima`, where ABC and ABC-Miner+ have the same predictive performance and almost the same model (size).

We used the matched-pair samples Wilcoxon Signed-Rank statistical test [9] to compare the predictive accuracies of ABC-Miner+ and ABC-Miner, where the samples are the datasets. According to the Wilcoxon test, the Z-value is -2.2012, and the p-value is 0.0139. Therefore, the results of ABC-Miner+ are statistically significantly better at the 5% significance level.

Table 1. Results of predictive accuracy (%), BN size (number of edges), and running time (in seconds) for the three ACO-based Bayesian classification algorithms.

	Predictive Accuracy			Model Size			Running Time		
Dataset	ABC-Miner	ABC	ABC-Miner+	ABC-Miner	ABC	ABC-Miner+	ABC-Miner	ABC	ABC-Miner+
`balace`	77.5	75.8	82.4	6.5	3.1	5.1	40	10	35
`breast-w`	92.7	90.4	95.8	23.6	8.3	14.8	540	130	280
`car`	97.2	86.7	98.1	14.5	6.0	11.2	480	240	310
`contraceptive`	66.5	75.0	76.7	18.7	7.2	10.4	220	150	550
`credit-a`	86.5	81.8	84.2	23.6	10.3	15.3	230	140	210
`credit-g`	71.7	70.2	73.8	28.2	11.2	18.5	520	120	500
`dermatology`	99.1	96.9	98.4	34.6	17.4	26.6	440	240	450
`glass`	93.3	87.6	91.3	11.5	3.0	4.1	140	90	110
`hayes-roth`	80.0	80.2	80.2	12.7	6.0	7.3	60	10	30
`heart-c`	83.4	74.6	86.9	18.4	10.2	21.3	510	250	480
`inf`	82.4	78.7	85.7	5.1	3.0	4.2	20	10	13
`ionosphere`	96.2	92.3	96.8	21.6	20.4	25.5	560	200	310
`monk`	65.2	74.2	74.2	17.4	6.3	6.3	30	10	50
`nursery`	98.2	94.7	96.7	22.6	6.0	14.6	430	290	320
`pima`	77.8	81.4	81.4	7.4	3.0	3.1	16	10	18
`soybean`	95.6	92.9	95.6	21.8	18.4	28.6	420	180	360
`tic-tac-to`	86.4	85.6	86.8	27.8	8.0	9.6	360	160	320
`vote`	94.8	94.3	95.6	34.6	12.4	15.6	400	280	280

In terms of model size, ABC obviously produces the smallest BN models, whose sizes are lower limits for the sizes of the models produced by ABC-Miner+, since the second phase of that algorithm can only add (and not remove) more edges to the BNs learnt by ABC. Moreover, the maximum number of the edges in a BN produced by ABC equals to the number of the input attributes (if all the input variables have relationships to the class), which is also the minimum number of edges that a BAN produced by ABC-Miner may have (if the local search procedure removed all the edges between the input variables and reduced the BAN to a Naïve-Bayes structure).

Besides, in terms of model size ABC-Miner+ outperformed the original ABC-Miner in all the datasets, producing BN classification models with fewer edges. The feature selection process implicitly performed by our extended algorithm can be easily noticed in the results of ABC. In some datasets, such as `breast-w`, `credit-a`, `credit-g`, and `dermatology`, the number of edges in the model produced by ABC-Miner+ is less than the number of input attributes in the corresponding dataset. This means that the produced BN classification model does not have all the input variables related to the class variable.

In terms of running time, as expected, ABC took the least amount of time to finish its execution in all the datasets. On the other hand, the two-phase ABC-Miner+ algorithm achieved a shorter execution time than ABC-Miner in 14 datasets. The reason behind that, as explained in Section 4, is that the first phase of the ABC-Miner+ algorithm reduced the search space for the second phase, after producing a BN structure with different dependency relationship types defined between the input and the class variables, and the first phase (ABC) does not consume a large amount of time, as shown in the results.

6 Concluding Remarks

In this paper, we have introduced ABC-Miner+ an extended version of an ACO algorithm for learning BN classifiers. ABC-Miner+ builds class Markov blanket-based BN classification models, in which it is not necessary to have an edge between the class variable and each of the input variables, and the edges between the class and the input variables may have different directions; unlike ABC-Miner, which learns BAN models. Empirical results showed that, overall, the ABC-Miner+ algorithm has an improved performance over the original ABC-Miner in terms of predictive accuracy, model size, and running time.

As a future research direction, we would like to investigate a different approach to learn the Markov blankets in a single integrated phase, rather than in two sequential phases as in ABC-Miner+. Moreover, we would like to try techniques to avoid over-fitting on the learning set during the training phase, like using different random partitions of learning\validation sets each iteration.

References

1. (UCI Repository of Machine Learning Databases Retrieved Oct 2011 from, URL:wwwicsuciedu/ mlearn/MLRepositoryhtml)

2. de Campos, L.M., Fernandez-Luna, J.M., Gamez, J.A., Puerta, J.M.: Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning 31(3), 291–311 (2002)
3. Cheng, J., Greiner, R.: Learning bayesian belief network classifiers: Algorithms and system. 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence pp. 141–151 (2001)
4. Chickering, D.M.: Learning Bayesian Networks is NP-Complete. Advanced Technologies Division, Microsoft Corporation, Redmond, WA, Technical Repor (1996)
5. Daly, R., Shen, Q., Aitken, S.: Learning bayesian networks: Approaches and issues. Knowledge Engineering Reviews 26(2), 99–157 (2011)
6. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004)
7. Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P., Smyth, P.: Bayesian Network Classifiers. Machine Learning 29, 131–163 (1997)
8. Heckerman, D.: A Tutorial on Learning with Bayesian Networks. Studies in Computational Intelligence: Innovations in Bayesian Networks 156, 33–82 (2008)
9. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press (2011)
10. Jiang, L., Wang, D., Cai, Z., Yan, X.: Survey of improving naïve-bayes for classification. 3rd International Conference on Advanced Data Mining and Applications (ADMA) pp. 134–145 (2007)
11. Kononenko, I.: Semi-naive bayesian classifier. In: The European working session on learning on Machine learning. pp. 206–219. EWSL-91, Springer-Verlag, New York, NY, USA (1991)
12. Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation 11, 651–665 (2007)
13. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: swarm intelligence for data mining. Machine Learning 82(1), 1–42 (2011)
14. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data Mining with an Ant Colony Optimization Algorithm. IEEE Transactions on Evolutionary Computation 6(4), 321–332 (2002)
15. Pearl, J.: Causality: Models, Reasoning and Inference. Cambridge University Press (2000)
16. Pinto, P.C., Nägele, A., Dejori, M., Runkler, T.A., Ao: Using a Local Discovery Ant Algorithm for Bayesian Network Structure Learning. IEEE Transactions on Evolutionary Computation 13(4), 767–779 (2009)
17. Salama, K.M., Abdelbar, A.M., Otero, F.E., Freitas, A.A.: Utilizing Multiple Pheromones in an Ant-based Algorithm for Continuous-Attribute Classification Rule Discovery. Applied Soft Computing 13(1), 667–675 (2012)
18. Salama, K.M., Abdelbar, A., Freitas, A.A.: Multiple Pheromone Types and Other Extensions to the Ant-Miner Classification Rule Discovery Algorithm. Swarm Intelligence 5(3-4), 149–182 (2011)
19. Salama, K.M., Freitas, A.A.: ABC-Miner: an Ant-based Bayesian Classification Algorithm. International Conference on Swarm Intelligence (ANTS) 7461, 2677–2694 (2012)
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 3rd edn. (2010)
21. Wu, Y., McCall, J., Corne, D.: Two novel Ant Colony Optimization approaches for Bayesian network structure learning. International Conference on Evolutionary Computation (CEC) pp. 1–7 (2010)